Effective and efficient location influence mining in location-based social networks

Saleem, Muhammad Aamir; Kumar, Rohit; Calders, Toon; Pedersen, Torben Bach

Published in:
Knowledge and Information Systems

DOI (link to publication from Publisher):
10.1007/s10115-018-1240-8

Publication date:
2019

Document Version
Accepted author manuscript, peer reviewed version

Noname manuscript No.
(will be inserted by the editor)

Towards Location Influence in Location-Based Social Networks ⋆

Muhammad Aamir Saleem · Rohit Kumar · Toon Calders · Torben Bach Pedersen

Received: date / Accepted: date

Abstract Location-based social networks (LBSN) are social networks complemented with location data such as geo-tagged activity data of its users. In this paper, we study how users of an LBSN are navigating between locations and based on this information we select the most influential locations. In contrast to existing works on influence maximization, we are not per se interested in selecting the users with the largest set of friends or the set of locations visited by the most users; instead, we introduce a notion of *location influence* that captures the ability of a set of locations to reach out *geographically* by utilizing their visitors as message carriers. We further capture the influence of these visitors on their friends in LBSNs and utilize them to predict the potential future location influence more accurately. We provide exact on-line algorithms and more memory-efficient but approximate variants based on the HyperLogLog and the modified-HyperLogLog sketch to maintain a data structure called *Influence Oracle* that allows to efficiently find a top-k set of influential locations. Experiments show that our new location influence

⋆ This paper is a significant extension of the conference paper [25].

M. A. Saleem
Aalborg University, Denmark
Universite Libre de Bruxelles, Belgium
E-mail: maas@cs.aau.dk

R. Kumar
Universite Libre de Bruxelles, Belgium
Universitat Politecnica de Catalunya
E-mail: rohit.kumar@ulb.ac.be

T. Calders
Universite Libre de Bruxelles, Belgium
University of Antwerp, Belgium
E-mail: toon.calders@ulb.ac.be

T. B. Pedersen
Aalborg University, Denmark
E-mail: tbp@cs.aau.dk

notion favors diverse sets of locations with a large geographical spread and that our algorithms are efficient, scalable and allow to capture future location influence.

Keywords [
location-based social networks; location influence; influence maximization; geographical spread]

1 Introduction

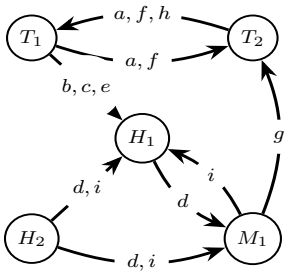
One of the domains in social network analysis [1, 10, 23, 26] that received ample attention over the past years is *influence maximization* [18], which aims at finding influential users based on their social activity. Applications like viral marketing utilize these influential users to maximize the information spread for advertising purposes [4]. With the pervasiveness of location-aware devices, nowadays, social network data is often complemented with geographical information. For instance, users of a social network share geo-tagged content such as locations they are currently visiting with their friends. These social networks with location information are called location-based social networks (LBSNs). In LBSNs, the location information offers a new perspective to view users' social activities. In this paper, we study navigation patterns of users based on LBSN data to determine *influential locations*. Where other works concentrate on finding *influential users* [30], *popular events* [31], or *popular locations* [33], we are interested in identifying sets of locations that have a large *geographical* impact. Although often overlooked, the geographical aspect is of great importance in many applications. This geographical information can be utilized to provide more targeted marketing strategies. For example, unlike viral marketing which focuses on finding influential users and spreading the message via word of mouth marketing (WOMM), influential locations can be found and information can be spread using out-of-home/ outdoor marketing (OOH) e.g., by putting advertisements on billboards and distributing promotional items on such locations. For instance, consider the following example.

Example 1 A marketer is interested in creating visibility for her products to the maximum regions in a city by offering free promotional items such as T-shirts with a printed promotional message. To do that she has to choose locations to distribute the promotional items to visitors.

In order to choose the most suitable locations for offering these items, not only the popularity of the places is important, but also the geographical reach. By visiting other locations, people that were exposed to the advertisement, especially the receivers of the promotional items, may indirectly promote the products. For example, by wearing the shirt they expose the T-shirt's message to the people at the places they go to and they talk about it with their friends and relatives. Thus, when the goal is to create awareness of the product name, it may be preferable to have a moderate presence in many locations throughout

Check-ins

loc	Users		
	t=1	t=2	t=3
T_1	b, c, e, f	a, h	f
T_2	a, h	f, g	a
M_1	g	i	d
H_1	–	b, c, d, e	i
H_2	d, i	–	–



Friendships

User	Friends
a	c, h, i
b	d, f, i
c	a, f
d	b, e
e	d, g
f	c, b
g	e
h	a
i	a, b

Fig. 1: [25] Running example of an LBSN: Check-ins(L) shows the visits of users (represented by lower-case letters; a, b, etc.) at locations (represented by upper-case letters; T_1, H_1, etc.) at time stamps t=1, 2 and 3. Graph (C) depicts the movement of users between consecutive locations. Friendships (R) show the friends of each user in the social network.

the whole city rather than a high impact in only a few locations. An illustration of this example is given in Figure 1. Nodes represent popular locations of different categories, such as tourist attractions (T_1, T_2), a metro station (M_1), and hotels (H_1 and H_2). Lowercase letters represent users. For each user, her friends in the social network and check-ins have been given. The top-2 locations with the maximal number of unique visitors are T_1 and M_1. The geographical impact of these locations, however, is not optimal; visitors of these locations reach only T_2 and H_1. On the other hand, the visitors of T_1 and H_2 visit all locations, i.e., users a, f and b, c, e visit T_2 and H_1 after visiting T_1, respectively, and users d, i visit H_1 and M_1 after H_2.

To capture geographical spread and influence, in Section 3, we introduce the notion of a *bridging visitor* between two locations as a user that visits both locations within a limited time span. If there is a significant number of bridging visitors from one location to another, we say that there is an influence. We introduce different models that capture when the number of bridging visitors is considered to be sufficient to claim influence between locations. One model is based on the absolute number of visitors, and one on the relative number. For each of these two models, we further present a direct bridging visitor based location influence model and two friendship-based location influence models that take the social graph of the LBSN into account. The friendship-based location influence models are based on the following observations obtained by detailed analysis of three real-life LBSNs. The first observation is that users tend to follow their friends and perform the same activities; e.g., in Figure 1, users i and f visited the same locations T_1 and H_1 after their friend b did. The second observation is that sometimes the number of visits/activities to some locations can be rather low because of data sparsity, especially when the time window used in the algorithms is small. The data sparsity may affect the location influence models and capture less influential seeds. Considering these observations, the friendship-graph-based models allow to compute potential

future bridging visitors. By incorporating such visitors, the models overcome the data sparsity problem and capture location influence more accurately. The first friendship-based influence model considers all friends of bridging visitors, while the second model computes the influence of bridging visitors on their friends in LBSNs and only incorporates the strongly influenced friends as potential future bridging visitors. Based on these models, we define influence for sets of locations and the *location influence maximization problem*: *Given an LBSN and a parameter k, find a set of k locations such that their combined location influence on other locations is maximal.*

To solve this problem, in Section 4 a data structure, called *Influence Oracle*, is presented that maintains a summary of the LSBN data that allows to determine the influence of any set of locations at any time. Based on this data structure, we can easily solve the location influence maximization problem using a greedy algorithm. As for large LBSNs with lots of activities the memory requirements of our algorithm can become prohibitively large, we also develop a more memory-friendly version based upon the well-known HyperLogLog sketch [11]. This algorithm gets further refined in Section 4.3 where we introduce a single-carrier-based influence maximization mechanism for capturing influence in information propagation scenarios where even a single carrier can carry the influence such as propagation of infections, and confidential information in specialized information networks. We provide off-line and on-line memory- and time-efficient algorithms for the single-carrier-based-influence maximization. Next, in Section 5 we provide a greedy algorithm for finding top-k influential locations.

In Section 6 we analyze several LBSNs to select reasonable threshold values for our models and to verify our claims. In Section 7 we evaluate the proposed notions and algorithms using the real-world datasets in term of effectiveness and efficiency.

In summary, the main contributions of this paper are (i) the introduction and motivation of a new location influence notion based on LBSN data, (ii) the development of an efficient Influence Oracle, and (iii) the demonstration of the usefulness of the location influence maximization problem in real-life LBSNs.

This paper is an extended version of the conference paper [25]. As an extension, we present a novel mechanism for spreading location influence that incorporates the influential users based on their geographical activities and social friends. The mechanism is given in Section 3.1.3. On the basis of the mechanism, we further propose two variants of absolute and relative influence models (given in Section 3.2). The new algorithms for finding such location influence are provided in Section 4.1. Furthermore, the single-carrier-based influence maximization mechanisms (given in Section 4.3) and the two algorithms for capturing such influence also constitute previously unpublished work. The methods for finding suitable values of the thresholds for new models are given in Section 6. We further present a set of new experiments for validating the proposed approaches in terms of effectiveness (Section 7.2) and

efficiency (Section 7.5) as well to evaluate their significance in comparison with existing methods.

2 Related Work

Influence maximization in the context of traditional social networks and LBSNs has been studied in much detail. We divide the existing studies in the domain into three groups. The first group consists of approaches for finding influential users in traditional social networks. The second group covers studies that use check-ins as an additional source of data to identify influential users in LBSNs, whereas the third group utilizes the check-ins for finding influential locations in LBSNs.

Influential Users in Social Networks. The influence maximization approaches in social networks are generally divided into two main groups. The first group of studies [8,24,18,6] operates on static graphs and assumes that the influence relationships among nodes are already known. They compute the influence probability of a node using probabilistic simulations and use them for determining influence among nodes. These approaches do not capture the temporal and dynamic nature of real networks such as social media. On the other hand, the second group of studies in this category [13,9,16,14] is data-driven and requires interactions of users and their activities. They compute influence probabilities based on relationships and historic activities of nodes such as common actions among two friends within a specified time. Thus, these studies are more suitable for dynamic networks such as LBSNs. Goyal et al. [14], propose the first data-based approach for finding influential users in social networks by considering the temporal aspect in the cascade of common activities of users. In [16], they further introduce a time window based approach to determine the true leaders in social networks. In [15], they present several models to compute influence probabilities. They provide static models based on likelihood estimation, as well as continuous and discrete time models for capturing the dynamic behavior of users in social networks. However, the limitations of these approaches are their assumptions that information propagation is non-cyclic and thus users can perform an action only once. In order to find the influence of users, we provide an extension of [15] as part of our influential friends-based location influence model. Our algorithm identifies influential nodes without any constraints on the number of times a user performs an action. It further allows cyclic propagation of information.

Influential users and events in LBSNs. Zhang et al. [31] use the social and the geographical correlation of users to find influential users and popular events. Users with many social connections are considered influential and events visited by them are considered important. Similarly, Wu et al. [30] identify influential users in LBSNs on the basis of the number of followers of their activities (check-ins). Li et al. [20] and Bouros et al. [2] on the other hand, identify regionally influential users on the basis of their activities. The focus of the work by Wen et al. [28] and Zhou et al. [32] is to find and utilize

the influential users for product marketing strategies such as word-of-mouth. Our focus, however, is to find influential *locations* that could be used, e.g., for outdoor marketing, hence, none of the previous works applies directly to our problem.

Location Promotion in LBSNs. Zhu et al. [33], Hai [17], and Wang et al. [27] study location promotion. Given a target location, their aim is to find the users that should be advertised to attract more visitors to this location. Doan et al. [7] compute the popularity ranks of locations based on the number of visitors. Zhou et al. [32] study the product promotion in O2O (on-line to off-line) model using LBSNs. Their model combines the on-line features, i.e., network topology (social network) and off-line user properties such as daily activity area and location preferences of users. Based on these features they find top-k users that can maximize the number of influenced users for a given location (product). These studies have different objectives as compared to our problem statement as they focus on finding top-k users that can attract the maximum visitor for a given location.

Novelty. Our work is different from all of the above as we focus on finding a *set of influential locations* where influence is defined using visitors as a mean to spread influence to other locations. Applications include outdoor marketing by selecting locations with the maximal geographical spread.

3 Location-Based Influence

In this section, we first provide preliminary definitions, then present location influence and different models to capture it, and finally we formally define the *Location Influence Maximization* and *Oracle* problems.

Let a set of users U and a set of locations L be given.

Definition 1 An *activity* [25] is a visit of a user to a location. It is a triplet (u, l, t), where $u \in U$ is a user, $l \in L$ a location and t is the time of visit of u to l. The set of all activities over U and L is denoted $\mathcal{A}(U, L)$.

Definition 2 A *Location-based Social Network (LBSN)*[25] over U and L consists of a graph $G_S(U, F)$, called the *social graph*, where $F \subseteq \{\{u, v\} | u, v \in U\}$ represents friendships between users, and a set of activities $A \subseteq \mathcal{A}(U, L)$. It is denoted $LBSN(G_S, A)$.

3.1 Bridging Visitors for Location Influence

We define the influence of a location by its capacity to spread its visitors to other locations. The intuition behind this is that the visitors exposed to a message at a location will spread the message to other locations they visit. Thus, the location influence (indirectly) captures the capability of a location to spread a message to other geographical regions by using common visitors as message carriers. The effect of an activity in a location, however, usually

remains effective only for a limited time. We capture this time with the *influence window* threshold ω. Such visitors that spread messages among locations based on their activities in LBSNs are called *Bridging Visitors (B)*. Next, we provide three types of bridging visitors for spreading location influence in LBSNs.

3.1.1 Direct Bridging Visitors

The first type of bridging visitors is called *Direct Bridging Visitors*:

Definition 3 Direct Bridging Visitor [25]: Given an $LBSN(G_S, A)$ and time window ω, a user u is said to be a *direct bridging visitor* from location s to location d if there exist activities $(u, s, t_s), (u, d, t_d) \in A$ such that $0 < t_d - t_s \leq \omega$. We denote the set of all direct bridging visitors from s to d by $B_D(s, d)$.

Example 2 Consider the running example of Figure 1. Let $\omega = 2$. Then, $B_D(T_1, H_1) = \{b, c, e\}, B_D(H_2, H_1) = \{d, i\}$ and $B_D(M_1, H_1) = \{i\}$.

3.1.2 Friends-Based Bridging Visitors

Activity data in LBSNs is often sparse in the sense that the number of check-ins per location is low. In Section 7, we see that the real-world datasets have only up to 6 check-ins per location on average. This sparsity of data affects the computation of location influence as usually there are very few bridging visitors among locations. In order to deal with this issue, we use the observation that users tend to perform similar activities as their friends (this claim is verified and confirmed in Section 6). Thus, the friends of bridging visitors have potential to carry the same message as the bridging visitors do. Based, on this observation we define *Friends-Based Bridging Visitors:*

Definition 4 Friends-Based Bridging Visitors: Given an $LBSN(G_S, A)$, time window ω, locations s and d, and direct bridging visitors from s to d $B_D(s, d)$, the set of *Friends-Based bridging visitors* between s and d is denoted by $B_F(s, d)$:

$$B_F(s, d) = B_D(s, d) \bigcup_{u \in B_D(s,d)} F_u \tag{1}$$

where F_u is the set of friends of u, i.e., $F_u = \{v | (u, v) \in F\}$

Example 3 Consider again the running example of Figure 1. Let $\omega = 2$. Then, $B_F(T_1, H_1) = \{a, b, c, d, e, f, g, i\}, B_F(H_2, H_1) = \{a, b, d, e, i\}$ and $B_F(M_1, H_1) = \{a, b, i\}$.

3.1.3 Influenced Friends-Based Bridging Visitors

Next, we further improve the notion of bridging visitors based on the following observation. Not all friends of bridging visitors may follow them, thus considering all of them as potential future visitors may bring high inaccuracy in predicting the number of bridging visitors and so in capturing the location influence. To improve the accuracy, we evaluate the ability of each bridging visitor to persuade their friends to follow them. The friends that are significantly influenced by the bridging visitors are called *Influenced Friends-Based Bridging visitors*.

A user v is considered to be influenced by a user u, if u visits a location l and v visits the same location after u within a particular time. In order to find such influence, we present an extended version of an existing algorithm given in [15] that computes the influence probabilities using a Bernoulli distribution based on partial credit distribution and discrete time constraint models [15]. According to the model, the influence probability is measured by the ratio of the number of successful attempts to persuade the influenced user to follow the influential user's activities over the total number of trials. Considering that a user can be influenced by multiple sources for an activity, the influential credit for each following activity is distributed among all such influential parents using the Partial Credit Distribution model. Furthermore, as influence probability is dependent on time, a discrete time constraint model is incorporated, which ensures that a user can influence other users only within the given time window. It is worth noting that such a time window can be different than the ω given in Definition 3. However, for our experiments, we consider the same ω for such a time window because we consider ω as a maximal time between two activities to still consider them connected. Goyal et al. in [15] do not capture repeating activities of users considering that traditional social network users are unlikely to repeat their actions such as re-posting the same contents. However, in LBSNs, users may visit the same locations again. This implies that, if an influential user visits a location after her influenced user, she is considered to be influenced by her influenced user. Our proposed algorithm (given in Algorithm 2) captures multiple activities of users at the same locations and thus, such aforementioned relationships. We ensure however that a user is influenced maximally once for an activity, i.e., a visit to a location, regardless of the number of times she visits the same location within ω. However, if the influential user visits the same location at another time, she may influence the same influenced user again. We denoted the influence probability of a user u on v as $p_{u,v}$. Next, we utilize such influence probabilities to define the influenced friends-based bridging visitors.

Definition 5 Influenced Friends-Based Bridging Visitors: Given $LBSN(G_S, A)$, time window ω, locations s and d, direct bridging visitors from s to d $B_D(s, d)$, and a threshold of the influence probability between users θ, a set of *Influenced Friends-Based bridging visitors* between s and d

is denoted by $B_I(s, d)$:

$$B_I(s,d) = B_D(s,d) \cup \{u \in U | \sum_{v \in B_D(s,d)} p_{v,u} \geq \theta\} \quad (2)$$

Example 4 Let $\tau_{IA} = 2$, $\omega = 2$ and $\theta = 0.2$. In the running example, a is the influenced user of h. a followed one out of two activities of h, i.e., for visiting T_2 and there is no other friends influencing a for this activity, thus, $p_{h,a} = 1/2 = 0.5 \geq \theta$. Similarly, the influenced users of b and c are $\{i, f\}$, and $\{a, f\}$, respectively. The other users do not have any influenced visitors as their influence probability is less than θ.

3.2 Methods for determining Location Influence

Next to the selection of message carriers, a second dimension is when we consider influence to be present and to what extent. For this purpose, we introduce two influence models (M).

3.2.1 Absolute Influence Model (M_A)

In practice, if a significant number of people perform an activity, then it is considered compelling. Thus, in order to avoid insignificant influences among locations, we use a threshold τ_A. The influence of a location s on a location d is considered only if the number of bridging visitors from s to d is greater than τ_A. This model is referred as the *Absolute Influence Model (M_A)*. The influence of a location s on d under M_A is represented by $I_A(s, d)$:

$$I_A(s,d) := \begin{cases} 1, & \text{if } |B(s,d)| \geq \tau_A \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

We instantiate B in Equation 3, with B_D, B_F or B_I, and τ_A with τ_{DA}, τ_{FA} or τ_{IA} to compute direct absolute influence (I_{DA}), friends absolute influence (I_{FA}) or influenced-friends absolute influence (I_{IA}), respectively.

Example 5 Consider the running example of Figure 1. Let the information carriers be the direct bridging visitors B_D, $\tau_A = 2$, and $\omega = 2$. Then, $I_{DA}(T_1, H_1) = 1$ because $|B_D(T_1, H_1)| = 3$. Similarly, $I_{DA}(H_2, H_1) = 1$. However, $I_A(M_1, H_1) = 0$ because $|B_D(M_1, H_1)| = 1$.

The influence between two locations may change with the value of τ_A and ω. For example, if we update the value of τ_A to 3 and ω to 2, $I_{DA}(T_1, H_1) = 1$, but, $I_{DA}(H_2, H_1)$ becomes 0.

3.2.2 Relative Influence Model (M_R)

In M_A, the influences of two pairs of locations are considered equal as long as the number of their bridging visitors is greater than τ_A. Sometimes, however, the relative number of contributed bridging visitors is important. Consider, for example, a popular location s that attracts many visitors and a non-popular location d with few visitors. In such a setting, to capture the influence of s on d, we may have to set the absolute threshold τ_A very low. This low value of τ_A, however, may result in many other popular locations being influenced by s, even if only a very small fraction of their visitors comes from s. Therefore, in such situations, it may be beneficial to use different thresholds for different destinations, relative to the number of visitors in these destination locations. This notion is captured by the *Relative Influence Model (M_R)*. The influence of s on d under M_R is represented by $I_R(s,d)$ and is parameterized by the relative threshold τ_R:

$$I_R(s,d) := \begin{cases} 1, & \text{if } \frac{|B(s,d)|}{|V(d)|} \geq \tau_R \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $V(d)$ is the set of users who visited location d. We instantiate B in Equation 4, with B_D, B_F or B_I, τ_R with τ_{DR}, τ_{FR} or τ_{IR}, and V with V_D, V_F: a set of V_D and their friends, or V_I: a set of V_D and influenced friends of visitors in V_D, to compute direct relative influence (I_{DR}), friends relative influence (I_{FR}) or influenced-friends relative influence (I_{IR}).

Example 6 Consider the running example given in Figure 1. Let the information carriers be the direct bridging visitors B_D, $\tau_I R = 0.4$, and $\omega = 2$. In this example, $I_{DR}(T_1, H_1) = 1$ because $\frac{|B_D(T_1,H_1)|}{|V_D(H_1)|} = \frac{|\{b,c,e\}|}{|\{b,c,d,e,i\}|} = \frac{3}{5} \geq \tau_{IR}$, Similarly, $I_R(H_2, H_1) = 1$ and $I_{DR}(M_1, H_1) = 0$.

In subsection 3.2, we presented two ways to determine the location influence. Each of the ways can utilize any of the three types of bridging visitors (given in subsection 3.1) to spread the location influence. Thus, in total, we have six models for spreading influence in LBSNs as given in Table 1.

3.3 Combined Location Influence

Based on the influence models, a location can influence multiple other locations. In order to capture such influenced locations, we define the *location influence set*:

Definition 6 Given a location s, and an influence model M, the *location Influence Set* $\phi_{I_M}(s)$ is the set of all locations for which the influence of s on that location under M is 1, i.e., $\phi_{I_M}(s) = \{d \in L \mid I_M(s,d) = 1\}$.

	Location-Based Influence Models					
	Absolute Influence			Relative Influence		
	Direct Bridging Visitors	**Friends-Based Bridging Visitors**	**Influenced Friends-based Bridging Visitors**	**Direct Bridging Visitors**	**Friends-Based Bridging Visitors**	**Influenced Friends-based Bridging Visitors**
Label	Direct Absolute Model (M_{DA})	Friends Absolute Model (M_{FA})	Influenced Friends Absolute Model (M_{IA})	Direct Relative Model (M_{DR})	Friends Relative Model (M_{FR})	Influenced Friends Relative Model (M_{IR})
Parameters	τ_{DA}, ω	τ_{FA}, ω	$\tau_{IA}, \omega, \theta$	τ_{DR}, ω	τ_{FR}, ω	$\tau_{IR}, \omega, \theta$
Location Influence	I_{DA}	I_{FA}	I_{IA}	I_{DR}	I_{FR}	I_{IR}

Table 1: Types of location-based influence models with labels and the parameters. Further, notations of the location influences captured by these models are also depicted.

Next, we define *combined location influence* for a set of locations S. To do this, we use the following principled approach: any activity at one of the locations of S is considered an activity from S. In that way we can capture the cumulative effect of the locations in S; even though all locations in S, in isolation may not influence a location d, together they may influence it. The bridging visitors from a set of locations S to d is represented by $B(S, d)$:

$$B(S, d) = \bigcup_{s \in S} B(s, d) \tag{5}$$

The influence of a set of locations S on location d under M_A and M_R is defined similarly as for single locations. For instance, the influence of S under M_A is given by

$$I_A(S, d) := \begin{cases} 1, & \text{if } |B(S, d)| \geq \tau_A \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Example 7 In Figure 1, let $\omega = 2$, $\tau_A = 3$ and $S = \{T_1, M_1\}$. Under M_A, $T_2 \notin \phi(T_1)$ and $T_2 \notin \phi(M_1)$. However, $T_2 \in \phi(S)$ as $|B(S, T_2)| = |\{a, f, g\}| \geq \tau_A$.

3.4 Problem Formulation

Based on these influence models, we now formally define the problem statements. We first present a problem statement for finding the most influential locations in LBSNs:

Problem 1 (Location Influence Maximization Problem) Given a parameter k, an $LBSN(G_S, A)$, and an influence model M, the location influence maximization problem is to find a subset $S \subseteq L$ of locations, such that $|S| \leq k$ and the number of influenced locations $|\phi_{I_M}(S)|$ is maximum.

In order to solve the location influence maximization problem efficiently, we first introduce an efficient solution to the following subproblem.

Problem 2 (Oracle Problem) Given an $LBSN(G_S, A)$ and an influence model M, construct a data structure that allows to answer: *Given a set of locations $S \subseteq L$ and a threshold τ, what is the combined location influence $\phi_{I_M}(S)$.*

4 Influence Oracle

In this section, we provide solutions for the Oracle problem. First, in Section 4.1, we provide a generic algorithm for constructing an influence oracle for any influence model. Then, in Section 4.2, we present an approximate but a more memory- and time-efficient algorithm for constructing the Influence Oracles for the $M_D(M_{DA}$ *and* $M_{DR})$ and the $M_F(M_{FA}$ *and* $M_{FR})$ models. After that, in Section 4.3 we present an even more efficient algorithm for the special case of the M_{DA} model with $\tau = 1$.

4.1 Exact Influence Oracle

In this section, we provide a data structure for maintaining exact location summaries for each location which works for the M_D model. Extension of this algorithm for incorporating the M_F and the M_I model are discussed later in this section.

Definition 7 The *Complete location summary* for a location $s \in L$ is the set of locations that have at least one bridging visitor from s, together with these bridging visitors; i.e., $\varphi(s) := \{(d, B(s,d)) \mid d \in L \wedge |B(s,d)| > 0\}$.

We assume activities arrive continuously and deal with them one by one. For the Oracle, we maintain a summary that consists of the collection of individual summaries $\varphi(s)$ for each location S. We present an on-line algorithm (Algorithm 1) to incrementally update these summaries.

If a user u visits a location s at time t, then u acts as a bridging visitor between all the locations u visited within the last ω time stamps and s. Therefore, for each user $u \in U$, we maintain a set of locations the user has visited and the corresponding latest visiting time. This is called the *visit history* $\mathcal{H}(u)$ and is defined as $\mathcal{H}(u) := \{(s, t_{max}) | u \in V(s), t_{max} = \max\{t \mid (u, l, t) \in A\}\}$. Suppose that we have the complete location summary for the check-ins so far and the visit history of all users, and a new activity (u, d, t) arrives. We update the complete location summary as follows: the location-time pair (d, t) is added in $\mathcal{H}(u)$ if d does not already appear in the visit history, otherwise the latest visit time of d is updated to t in $\mathcal{H}(u)$ (line 13). Furthermore, for every other location-latest visit time pair (s, t') in the history of u, $\varphi(s)$ is updated by adding user u to the set of bridging visitors from s to d provided that the difference between the time stamps $t - t'$ does not exceed the threshold ω (line $5 - 10$). This procedure is illustrated in Algorithm 1. The visit history $\mathcal{H}(u)$ is pruned at line 11 to remove those locations which were visited by u more

Algorithm 1: Exact Oracle: Updating complete location summaries [25]

Input: New activity (u, d, t); threshold ω; $\forall l \in L$, $\varphi(l)$ is given; $\mathcal{H}(u)$
Output: Updated $\varphi(.)$ and $\mathcal{H}(.)$
begin
  foreach $(s, t^{'}) \in \mathcal{H}(u)$ **do**
    if $t - t^{'} \leq \omega$ **then**
      if $\exists (d, B(s,d)) \in \varphi(s)$ **then**
        replace $(d, B(s,d)) \in \varphi(s)$ with $(d, B(s,d) \cup \{u\})$
      else
        add $(d, \{u\})$ to $\varphi(s)$
    else
      $\mathcal{H}(u) \leftarrow \mathcal{H}(u) \setminus \{(s, t^{'})\}$; `// Too old to be a bridging visitor`
  if $\exists (d, t') \in \mathcal{H}(u)$ **then**
    replace (d, t') with (d, t)
  else
    add (d, t) to $\mathcal{H}(u)$

than ω time ago. Pruning of old locations from the visit history can be done at regular interval for all locations.

	$t=1$	$t=2$	$t=3$	t=5
Activity:	$(i, H_2, 1)$ $(d, H_2, 1)$	$(i, M_1, 2)$ $(d, H_1, 2)$	$(i, H_1, 3)$ $(d, M_1, 3)$	$(d, H_2, 5)$
$\mathcal{H}(i):$	$\{(H_2, 1)\}$	$\{(H_2, 1),$ $(M_1, 2)\}$	$\{(H_2, 1),$ $(M_1, 2),$ $(H_1, 3)\}$	$\{(H_1, 3)\}$
$\mathcal{H}(d):$	$\{(H_2, 1)\}$	$\{(H_2, 1),$ $(H_1, 2)\}$	$\{(H_2, 1),$ $(H_1, 2),$ $(M_1, 3)\}$	$\{(M_1, 3),$ $(H_2, 5)\}$
$\varphi(H_1):$	$\{\}$	$\{\}$	$\{(M_1, \{d\})\}$	$\{(M_1, \{d\})\}$
$\varphi(H_2):$	$\{\}$	$\{(H_1, \{d\}),$ $(M_1, \{i\})\}$	$\{(H_1, \{d\}),$ $(M_1, \{i, d\})\}$	$\{(H_1, \{d\}),$ $(M_1, \{i, d\})\}$
$\varphi(M_1):$	$\{\}$	$\{\}$	$\{(H_1, \{i\})\}$	$\{(H_1, \{i\}),$ $(H_2, \{i\})\}$

Fig. 2: [25] An example of updating locations summaries for location H_1, H_2 and M_1 and visit histories of users i and d under M_A model for $\omega = 2$ at every time stamp.

Example 8 We illustrate the algorithm using the running example shown in Figure 1. For simplicity, we only consider the activities of two users: d and i. We also add a new activity of d at H_2 at time stamp 5. In this example, we consider $\omega = 2$. The activities are processed one by one in increasing order of time. We show how the visit history $\mathcal{H}(i)$, $\mathcal{H}(d)$ and the complete location summaries $\varphi(H_1)$, $\varphi(H_2)$, $\varphi(M_1)$ evolve with different activities at different timestamps in Figure 2. Note, at time stamp 5 only $\varphi(M_1)$ is updated even

though M_1 and H_1 are both in the visit histories of d because $\omega = 2$. The visit history of d is pruned by removing H_1 from the $\mathcal{H}(d)$ as no future activities by d affect $\varphi(H_1)$. The visit time of H_2 is updated to the latest visit time. Similarly, $\mathcal{H}(i)$ is also pruned.

It can be observed from the example that a new activity of a user u only updates the complete location summary of the locations in the recent visit history of u. Notice that, since the activities of a user arrive in strictly increasing order of time, the size of $\mathcal{H}(u)$ is upper bounded by ω, as only locations that are visited within a time window ω are processed and a user can only visit one location at a time.

Proposition 1 *For the M_{DA} model, the time required to process an activity by Algorithm 1, is $\mathcal{O}(\omega \log(|U|))$. The complete location summary $\{\varphi(l)|l \in L\}$ can be stored in $\mathcal{O}(|L|^2|U|)$ memory and the visit history$\{\mathcal{H}(u)|u \in U\}$ in $\mathcal{O}(|U|\omega)$ memory.*

Proof The visit history $\mathcal{H}(u)$ for a user u can at maximum have ω locations hence the for loop in line 4 of the algorithm will run for maximum ω iteration. The maximum set size of the bridging visitors is $|U|$, so adding an element to the set will take maximum $\log(|U|)$ time using an appropriate data structure, such as a balanced tree for storing a set. Thus, the total time for processing an activity in the worst case is $\mathcal{O}(\omega \log(|U|))$. The memory complexity is straightforward as there could be maximally $|L|$ influenced locations and the bridging visitor set size is at most $|U|$, hence, the memory complexity is $\mathcal{O}(|L||U|)$ in the worst case for a location hence for all locations it is $\mathcal{O}(|L|^2|U|)$. □

Proposition 2 *For the M_{DA} model, the time required to produce $\phi(S)$ from $\{\varphi(l)|l \in S\}$ for given threshold τ and set of locations S is $\mathcal{O}(|S||L||U|)$.*

Proof Every location can have influence on maximally $|L|$ locations with the bridging visitor set size at most $|U|$. Hence, to produce $\phi(S)$, the union of sets of size $|U|$ has to be taken at most $|S||L|$ times, thus, the time complexity is $\mathcal{O}(|S||L||U|)$. □

Extending for Relative models. For the relative models, we additionally have to maintain the total number of unique visitors per location, which can be done in the worst case time $O(log(|U|))$ and space $O(|U|)$ per location and hence does not affect the overall complexity.

Extending for Friends based bridging visitors. For this scenario, we assume the friendship graph is given as an adjacency list $< u, u_{friends} >$ indexed by u. Hence, whenever we add u in the bridging visitor set (line 7 and 9 in Algorithm 1), we just have to add all the friends of the user u in the set of bridging visitors. As the number of friends is bounded by $|U|$, we get:

Proposition 3 *For Algorithm 1, the time required to process an activity for Friends based bridging visitors is $\mathcal{O}(\omega|U|)$. The memory required to maintain the summary is the same as for the M_D, $\mathcal{O}(|L|^2|U|)$.*

Algorithm 2: Influence Probabilities among users

```
1  Input: List of activities A, time threshold ω, Friendships F
2  Output: influenceEdges
3  begin
4      influenceEdges = Map() , userActs = Map() , influenceActs = Map()
5      A_g(l)= Group activities by location and sort by time
6      foreach A(l) ∈ A_g(l) do
7          currentSet = φ
8          followersSet = φ
9          foreach (u, l, t_u) ∈ A(l) do
10             increment userActs(u)
11             parents = φ
12             foreach v : (v, l, t_v) ∈ currentSet & (u, v) ∈ F do
13                 if (v, u, t_v) ∉ followerMap then
14                     if t_u − t_v ≥ ω then
15                         increment influenceActs(v → u)
16                         add (v, u, t_v) to followersSet
17                         add v to parents
18             foreach v ∈ parents do
19                 update influenceEdges < (v, u), p_{v,u} >
20         add (u, l, t_u) in currentSet
```

Proof Every user can have maximally $|U|$ friends and hence adding them in the bridging visitor set would take $|U|$ time. There are maximum ω location in the visit history of a user, thus, bridging visitors of ω locations would be updated giving a total time complexity of $\mathcal{O}(\omega|U|)$. □

Extending for Influenced Friends based bridging visitors. In this case, we first compute the influence probabilities of users among each other. The influence probabilities are computed using the algorithm 2. In this algorithm, first, we initialize $influenceEdges$ that stores the influence probabilities for each pair of influential and influenced users, $userActs$ that maintains the count of activities for users, and $influenceActs$ that tracks the number of influential activities of users among each other. We then group the activities based on locations (Line 5). We iteratively process all the activities that are performed at a location (line 9). In line 13, we ensure that a user is not influenced multiple times by the same activity. We consider the activities influential if they the time difference in following the activities is less than the window threshold (line 14). In lines 18-19, we compute/update the influence probability by which a user is influenced using the Bernoulli equation based on the number of influential activities, all activities and the influential users. The influence probabilities are stored in a hash-map.

Next, we use the influence probabilities for adding the influenced friends of bridging visitors for every pair of locations (influential-influenced locations). The pseudo code for the algorithm is given in Algorithm 3. It is worth noting that this is an off-line algorithm as we need to process all the activities first using Algorithm 1. After that, we process a complete bridging visitor set of a location pair at a time(line 6-16 in Algorithm 3). To do that, for each user in a bridging visitor set, we first fetch her influenced users with corresponding

Algorithm 3: Exact Oracle for Influenced Friends: Updating complete location summaries

```
1  Input: A all activities, ω time window, θ minimum influence threshold, F
     friendships
2  Output: Complete location summary φ(l) for all l ∈ L
3  begin
4      InfluenceEdges= Influence Probabilities among users(A, ω, F) ;
         // Algorithm 2
5      Run Algorithm 1 ∀(u, d, t) ∈ A ;     // This will generate φ(l) for all l ∈ L
6      foreach l ∈ L do
7          foreach s ∈ φ(l) do
8              InfluencedFriends ← φ
9              InfluentialBVs ← φ
10             foreach (u, v, P_{u,v}) ∈ InfluenceEdges do
11                 if x ∈ B(l, s) then
12                     add (u, v, p_{u,v}) to InfluentialBVs
13             foreach v : (u, v, P_{u,v}) ∈ InfluentialBVs do
14                 if (Sum(P_{u,v}), ∀u : (u, v, p_{u,v}) ∈ InfluentialBV) ≥ θ then
15                     add v to InfluencedFriends
16             B(l, s) = B(l, s)∪ influenced friends
```

influence probabilities (lines: 10-12). Then, we compute the cumulative influence probability for each influenced user by adding the influence probabilities of the influenced user with her every influential user in the bridging visitor set. The influenced users with cumulative influenced probability greater than the minimum influence threshold (θ) (given in Equation 2) are added in the set of bridging visitors (lines: 13-16). The same procedure is followed for adding influenced friends for V_D in the case of M_R.

Proposition 4 *For Algorithm 1, the space complexity for M_I is the same as for M_F i.e, $\mathcal{O}(|L|^2|U| + \omega + |U|^2)$, and the time required to compute influence oracle for the influenced friends-based model is $\mathcal{O}(\omega|U|\log(|U|)|A| + |L|^2|U|^2 + |A|(|A| + |U|))$.*

Proof A user can at most influence all of her friends which is equivalent to adding all friends of users in a bridging visitor set, as we do in M_F. Thus, the space complexity for M_I is same as for M_F, i.e., $\mathcal{O}(|L|^2|U| + \omega + |U|^2)$.

For computing influence oracle for M_I, we further need to find the influenced friends of bridging visitors. Thus, we first need to compute the influence probabilities among users. To do that we group the activities on the basis of location and then sort the activities performed at a location in a chronological order. Then, for each location, we iteratively consider all the activities to evaluate the influence relationship of users who performed them among each other and update their corresponding influence scores. As each activity is evaluated with every other activity thus in total we have $|A| * |A|/2$ iterations. We further traverse all influential users to assign them their credit, which can at most be $|U|$. The influence probabilities are computed once and stored in a hashmap. To add influenced friends in a bridging visitor set, we need to fetch the influenced friends and influence probabilities for every user

	Exact			**Approx**		
	Memory	Oracle Time	Query Time	Memory	Oracle Time	Query Time
M_{DA}	$O(\|U\|(\|L\|^2+\omega))$	$O(\omega \log(\|U\|)\|A\|)$	$O(\|S\|\|L\|\|U\|\|U\|)$	$\mathcal{O}(\|L\|^2 b + \|U\|\omega)$	$O(\omega\|A\|)$	$O(\|S\|\|L\|b)$
M_{DR}				$\mathcal{O}(\|L\|^2 b + \|U\|\omega + \|L\|\|U\|)$		
M_{FA}	$O(\|U\|(\|L\|^2+\omega + \|U\|))$	$O(\omega\|U\|\log(\|U\|)\|A\|)$		$\mathcal{O}(\|L\|^2 b + \|U\|\omega + \|U\|^2)$	$O(\omega\|U\|\|A\|)$	
M_{FR}				$\mathcal{O}(\|L\|^2 b + \|U\|\omega + \|U\|^2 + \|L\|\|U\|)$		
M_{IA}		$O(\omega\|U\|\log(\|U\|)\|A\| + \|L\|^2\|U\|^2 + \|A\|(\|A\|+\|U\|))$		N/A		
M_{IR}						

Table 2: Summary of time and space complexities for the influence models.

in the bridging visitor. The time to fetch the influenced visitors is constant. Thus, the time to add the influence visitors for a set of bridging visitors which at most can be $|U|$ is $|U| * |U|$ and thus, for each location pair, it is $|L|^2|U|^2$. This makes the overall time complexity for computing oracle for M_I as $\mathcal{O}(\omega|U|\log(|U|)|A| + |L|^2|U|^2 + |A|(|A| + |U|))$. □

4.2 Approximate Influence Oracle for M_D and M_F

In the worst case the memory requirements of the exact algorithm presented in the last section are quite stringent: for every pair of locations (s, d), in $\varphi(s)$ the complete list of bridging visitors from s to d is kept. Therefore, here we present an approximate algorithm for maintaining the complete location summaries in a more compact form. This compact representation leads to a significant saving especially in those cases where the window size ω is large since in that case the number of bridging visitors increases.

We observe that when computing the number of bridging visitors between s and d we do not need the exact set of bridging visitors between s and d, but only the cardinality of that set. For the relative number of bridging visitors, we additionally need only the numbers of visitors $|V(s)|$. Furthermore, as per Equation 5, in order to find the accumulated complete location summary, we need to combine two complete location summaries; for instance: the complete location summary $\varphi(\{s_1, s_2\})$ is obtained by taking the following pairwise union of $\varphi(s_1)$ and $\varphi(s_2)$: if $\varphi(s_1)$ and $\varphi(s_2)$ respectively contain the pairs $(d, B(s_1, d))$ and $(d, B(s_1, d))$, then $\varphi(\{s_1, s_2\})$ contains $(d, B(s_1, d) \cup B(s_2, d))$. But then again, for further computations, we only need the cardinality of the bridging visitor sets. Hence, if we accept approximate results, we could replace the exact set $B(s, d)$ with a succinct sketch of the set that allows to take unions and get an estimate of the cardinality of the set. Please note that we can approximate the bridging visitor set $B(s, d)$ only for the direct and friends-based bridging visitor sets. For the Influenced Friends based bridging visitor set, we need the exact set as we need to know all the users in $B(s, d)$ to find the set of Influenced friends. Therefore, the approx algorithm is only for the direct and friends-based influence models.

In our approx algorithm, we use the HyperLogLog sketch (HLL) [11] to replace the exact sets $B(s,d)$ and $V(s)$. The HLL sketch is a memory-efficient data structure of size 2^k that can be used to approximate the cardinality of a set by using an array. The constant k is a parameter which determines the accuracy of the approximation and is in our experiments in the order of 6 to 10. Furthermore, the HLL sketch allows unions in the sense that the HLL sketch of the union of two sets can be computed directly from the HLL sketches of the individual sets. For our algorithm, we consider the HLL algorithm as a black box. By using HLL, we not only reduce memory consumption but also improve computation time, because adding an element in an HLL sketch can be done in constant time and taking the union of two HLL sketches takes time $\mathcal{O}(2^k)$; that is: the time to take the union of two sets is independent of the size of the sets.

Proposition 5 *Let $b = 2^k$ be the size of the HLL sketch. For the M_D and M_F models, the time needed to process an activity using the HLL sketch to maintain $B(s,d)$ is $\mathcal{O}(\omega)$. The memory required to maintain the complete location summary $\{\varphi(l)|l \in L\}$ is $\mathcal{O}(|L|^2 b)$. The memory requirement for the visit history $\{\mathcal{H}(u)|u \in U\}$ will remains $\mathcal{O}(|U|\omega)$ as in the exact algorithm mentioned in Proposition 1.*

Proof Adding an element in a HLL set takes constant time, hence, to process the activity HLL set of ω locations will be updated in $\mathcal{O}(\omega)$. The size of the HLL set is b irrespective of the number of elements in the set and thus, the memory required to store $\varphi(l)$ is $\mathcal{O}(|L|b)$. Hence, for all locations the memory required is $\mathcal{O}(|L|^2 b)$. □

4.3 Single-Influencer based Influence Oracle

In this subsection, we go one more step further and develop an even more efficient algorithm for a very special case. In real life, there may be situations in which even one information carrier can spread information among locations. Examples may include infections or information items in highly specialized information networks with confidential information. Moreover, LBSN data is often sparse, thus, usually, has a very low number of influence carriers. These situations may also have been created artificially by lumping together multiple traces for reasons of privacy; in such a situation a single visit trace may actually correspond to multiple visitors. Thus, in such situations, we may have to rely on single carriers as a proxy for larger unobserved streams of people.

In this section, for this special case, we provide two approximate but more efficient algorithms; an on-line algorithm called *On-Sin* and an off-line but far more efficient algorithm called *Off-Sin* for solving the influence oracle problem.

4.3.1 On-Sin approach

The **On**-line **S**ingle influencer based **in**fluence oracle (*On-Sin*) approach is based upon the simple observation that for $\tau = 1$, we do not need to maintain

	$t=5$	$t=3$	$t=2$	t=1
Activity:	$(d, H_2, 5)$	$(i, H_1, 3)$ $(d, M_1, 3)$	$(i, M_1, 2)$ $(d, H_1, 2)$	$(i, H_2, 1)$ $(d, H_2, 1)$
$\mathcal{H}_f(d):$	$\{(H_2, 5)\}$	$(H_2, 5)$, $(M_1, 3)$	$\{(M_1, 3)$, $(H_1, 2)\}$	$\{(M_1, 3)$, $(H_1, 2)$, $(H_2, 1)\}$
$\mathcal{H}_f(i):$	$\{\}$	$\{(H_1, 3)\}$	$\{(H_1, 3)$, $(M_1, 2)\}$	$\{(H_1, 3)$, $(M_1, 2)$, $(H_2, 1)\}$
$\phi(H_1):$	$\{\}$	$\{\}$	$\{M_1\}$	$\{M_1\}$
$\phi(H_2):$	$\{\}$	$\{\}$	$\{\}$	$\{H_1, M_1\}$
$\phi(M_1):$	$\{\}$	$\{H_2\}$	$\{H_2, M_1\}$	$\{H_2, M_1\}$

Fig. 3: An example of updating location influence set for locations H_1, H_2 and M_1 for $\tau = 1$ and $\omega = 2$ by processing data in reverse order of time using Off-Sin algorithm.

the bridging visitor set $V_B(s, d)$ for locations s and d, because even one visitor implies influence and hence the location influence set $\phi(s)$ can be directly maintained by adding d whenever a user u visits d within ω time after visiting s. Hence, in the special case we do not need to maintain the complete location summary $\varphi(s)$ and directly maintain $\phi(s)$.

Furthermore, in order to find the most influential locations, we just need the cardinality of the location influence summary. Hence, we can replace $\phi(s)$ by a HyperLogLog(HLL) sketch. The memory required to store $\{\phi(l)|l \in L\}$ is $\mathcal{O}(|L|b)$ as for every location we just keep a HLL sketch. Please note that the time required to process an activity still remains the same at $\mathcal{O}(\omega)$ as in the worst case we still need to iterate over ω locations in the user visit history. Next, we provide an approach in which we could actually store an approximation of the user visit history to provide tremendous speedups.

4.3.2 Off-Sin approach

The **Off**line **S**ingle influencer based **in**fluence oracle (*Off-Sin*) algorithm is based on the observation that while processing an activity (u, s, t), if we know all the future locations u will visit during time t to $t + \omega$, then we can directly add those locations in the location influence set $\phi(s)$. In order to achieve this, we process all the activities in reverse order of time. As we are going reverse in time we cannot run this algorithm incrementally for new activities hence it is an off-line algorithm. A simple case is shown in the following example.

Example 9 Consider the activities of users i and j given in Figure 1. We process the activities of these users in reverse order of time. Figure 3 shows the update of $\phi(l)$ and $\mathcal{H}_f(u)$ for each activity. For sake of understanding, we represent the exact sets in the example but for the efficient algorithm the sets $\phi(s)$ and $\mathcal{H}_f(u)$ are approximated with HLL and vHLL sets respectively. At time $t = 3$,

Algorithm 4: Off-Sin: Location Influence summary by using modified HLL for $\tau = 1$

```
1 Input: Activity list A.
2 Output: φ(s) ∀s ∈ L
3 begin
4     Sort A in decreasing order of time.
5     φ(s) ← HLL   ∀s ∈ L
6     u_f ← vHLL   ∀u ∈ U
7     foreach (u, s, t) ∈ A do
8         φ(s) ← φ(s) ∪ subset(H_f(u), t, ω)
9         H_f(u).add(s, t)
```

location H_2 is added into influence set $\phi(M_1)$ as $(H_2, 5)$ was in $\mathcal{H}_f(d)$ and hence is within time window 2. $\mathcal{H}_f(d)$ and $\mathcal{H}_f(i)$ is updated as well. Note at time $t = 2$, $\mathcal{H}_f(d)$ is pruned and $(H_2, 5)$ is removed as it is out of window from current time.

Now, instead of the visit history $\mathcal{H}(u)$, we maintain the future visit history represented by $\mathcal{H}_f(u)$. At time t, $\mathcal{H}_f(u) = \{(s, t') | t' \geq t,\ t' - t \leq \omega, (u, s, t') \in A,\ \nexists t'' : t \leq t'' < t' \wedge (u, s, t'') \in A\}$. That is, the future visit history $\mathcal{H}_f(u)$ of a user u, maintains every location a user u visits in future and the earliest time in future the user visits that location. We can see that adding all locations $\mathcal{H}_f(u)$ that will be visited by u in $\phi(s)$ is much more efficient than adding s to $\phi(d)$ for all locations d in $\mathcal{H}(u)$. This is because now for every activity (u, s, t), instead of updating summaries of all locations in $\mathcal{H}(u)$ we need to just update the summary of s by merging it with $\mathcal{H}_f(u)$. Furthermore, we do not need the individual locations anymore in history, but only their cardinality. Thus, we can approximate the set $\mathcal{H}_f(u)$.

Now, while processing an activity (u, s, t), we update $\phi(s)$ and add all the locations, d in $\mathcal{H}_f(u)$ for which $\mathcal{H}_f(u) - t \leq \omega$. We do not need to iterate over the elements of the set $\mathcal{H}_f(u)$ but just need a subset of $\mathcal{H}_f(u)$ to get elements added during current time and a time window ω. Using a versioned HyperLogLog sketch (vHLL) [19], we can achieve such a time window based approximate set with much less memory and time requirements. vHLL is an extension of the HLL data structure which approximates a set and allows to get the cardinality of the set based on a specified time window. While adding an element, vHLL also maintains the time of addition of the element. vHLL provides a *subset* function which takes time t and window ω as an input and produces an HLL representation of a set. This set consists of elements that were added in vHLL during time t and $t + \omega$.

Algorithm 4 represents the *off-Sin* batch algorithm. We go through activities list A in reverse order. We treat the activities with later time stamps working our way from the end to the start of the log. The functions $subset(\mathcal{H}_f(u), t, \omega)$ at line 8 and $\mathcal{H}_f(u).add(d, t)$ at line 9 are functions provided by vHLL.

	Off-Sin			On-Sin		
	Memory	**Time Oracle**	**Time Query**	**Memory**	**Time Oracle**	**Time Query**
M_A	$\mathcal{O}(b(\lvert L\rvert + \lvert U\rvert \log\omega))$	$O(b\log(\omega)\lvert A\rvert)$	$O(\lvert S\rvert b)$	$\mathcal{O}(b\lvert L\rvert + \lvert U\rvert\omega)$	$O(\omega\lvert U\rvert\lvert A\rvert)$	$O(\lvert S\rvert b)$
M_R	$\mathcal{O}(b(\lvert L\rvert + \lvert U\rvert \log\omega) + \lvert L\rvert\lvert U\rvert)$			$\mathcal{O}(b(\lvert L\rvert + \lvert U\rvert \log\omega) + \lvert L\rvert\lvert U\rvert)$		

Table 3: Time and Space complexities for Single influencer-based Influence.

Proposition 6 *The time required to process an activity by Off-Sin in Algorithm 4 is $\mathcal{O}(b\log(\omega))$. The memory requirements to maintain $\mathcal{H}_f(u)$ improves from $\mathcal{O}(\omega)$ to $\mathcal{O}(b\log(\omega))$.*

Proof The time required to process an activity will be equal to the time required to find subset in line 8 and then updating $\mathcal{H}_f(u)$ at line 9. According the time complexity of vHLL given by Kumar et al. in [19], the add function takes $\mathcal{O}(\log\omega)$ time and the subset function takes $\mathcal{O}(b\log(\omega))$ time. The subset function returns a HLL sketch and $\phi(s)$ is also a HLL sketch, the union of two HLL sketches takes $\mathcal{O}(b)$ time. Hence, the total time complexity of Algorithm 4 is $\mathcal{O}(b\log(\omega) + \log(\omega) + b) = \mathcal{O}(b\log(\omega))$. $\mathcal{H}_f(u)$ is a vHLL set and as given by Kumar et al. in [19] the memory required by a vHLL set is $\mathcal{O}(b\log(\omega))$. □

5 Location Influence Maximization

In this section, we show that the influence oracle can be used for finding the most influential locations. We utilize the influence oracle and apply the standard greedy algorithm to compute top-k as obtaining an exact solution is intractable as the next proposition states.

Proposition 7 *The following problem is **NP**-hard for all influence models: given an LBSN and bounds k and β, does there exist a set of locations S of size k such that $|\phi(S)| \geq \beta$.*

Proof **NP**-hardness follows from a reduction from set cover. Consider an instance $\mathcal{S} = \{S_1, \ldots, S_m\}$ with all $S_i \subseteq \{1, \ldots, n\}$ and bound k of the set cover problem: does there exist a subset $\mathcal{S}'$ of $\mathcal{S}$ of size at most k such that $\bigcup\mathcal{S}' = \{1, \ldots, n\}$. We reduce this instance to a LBSN as follows: $L = \{l_1, \ldots, l_n\} \cup \{s_1, \ldots, s_m\}$, $U = \{u_1, \ldots, u_m\}$, $F = \emptyset$, $A = \{(u_i, s_i, 0) \mid i = 1 \ldots m\} \cup \{(u_i, l_j, j) \mid i = 1 \ldots m, j \in S_i\}$. That is, every element j of the domain $\{1, \ldots, n\}$ is associated to a location l_j, and for every set S_i we introduce a location s_i visited by user u_i at time 0. Furthermore, user u_i visits all locations l_j such that $j \in S_i$ at time stamp j. If we use the absolute model with $\tau = 1$ and $\omega \geq n + 1$, for $i = 1 \ldots m$, $\phi(\{s_i\}) = \{l_j \mid j \in S_i\}$. As such there exists a set cover of size k if and only if there exists a set of locations S of size k such that $|\phi(S)| = n$. □

Recall that the influence of a set of locations S is computed by accumulating the effect of all locations in S. It is hence possible that two locations s

and s' separately do not influence a target location d because individually they have too few bridging visitors to d, but together they reach the threshold. This situation occurs for instance in Figure 1, for the locations H_2 and M_1. These locations individually do not reach the threshold to influence H_1 for $\tau_A = 2$ and $\omega = 1$. However, together they do. One inconvenient consequence of this observation is that the influence function that we want to optimize is not submodular [22]. Indeed, in the example above, adding H_2 to the set $\{M_1\}$ gives a higher additional benefit (1 more influenced location) than adding H_2 to $\{\}$. Therefore, we do not have the usual guarantee on the quality of the greedy algorithm for selecting the top-k.

The main reason that we do not have the guarantee is that the benefit is not gradual; before the threshold is reached it is 0, after the threshold is reached it is 1. This means that a location that has $\tau - 1$ bridging visitors to 1000 other locations each, gives the same benefit as a location that does not have any bridging visitors. Clearly, nevertheless, the first location is more likely to lead to a good solution if later on additional locations are selected. Therefore, we would like to incorporate potential future benefits into our objective function. Thus, in order to compute the influence of a location, we consider locations that are influenced as well as those locations that are not yet influenced but have potential to be so in future. To characterize the potential of future benefit in combination with the number of influenced locations, we use the following formula:

$$LI(S) = (1 - \alpha) \times |\phi(S)| + (\alpha) \times \sum_{d \in L - S} (\min\{|B(S, d)|, \tau\}) \qquad (7)$$

In this formula, $\alpha = [0, 1]$ represents a trade-off between the number of influenced locations and a reward for potentially influenced locations. For relative models, we replace the $|B(S, d)|$ with $|B(S, d)|/|V(d)|$.

Next, we apply a greedy method on the basis of location influence to find top-k locations. We start with an empty set S of locations and iteratively add locations to it until we reach the required number of top elements: k. We start with an empty set S of locations and iteratively add locations to it until we reach the required number of top elements: k. In each step, for each location $s \in L$, we evaluate the effect of adding s to S, and keep the one that gives the highest benefit $LI(S)$. Then, we update $S \leftarrow S \cup \{l\}$.

Example 10 Consider the case in Figure 2 for $\omega = 1$, $\varphi(H_2) = \{(H_1, \{d\}), (M_1, \{i\})\}$, $\varphi(M_1) = \{(H_1, \{i\})\}$ and $\varphi(H_1) = \{(M_1, \{d\})\}$. We aim to find top-2 locations in this example with $\alpha = 0.1$ and $\tau = 2$. During the first iteration, $LI(H_2) = 0.9 \times 0 + 0.1 \times (1 + 1) = 0.2$, because H_2 does not completely influence any other location, however H_1 and M_1 are potentially influenced locations for the bridging visitors d and i, respectively. Similarly, $LI(M_1) = 0.1$ and $LI(H_1) = 0.1$. Thus, we choose H_2 as first seed as it has maximum value. In the next iteration, we first combine the seed H_2 with M_1 and compute the combined influence. Here, $LI(\{H_2, M_1\}) = 0.9 \times 1 + 0.1 \times (2) = 1.1$. Similarly, $LI(\{H_2, H_1\}) = 1.1$.

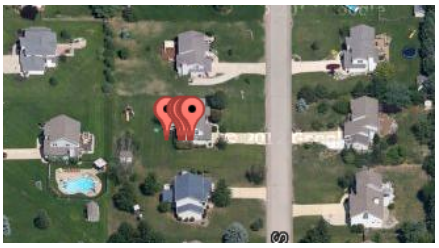


Fig. 4: [25]An example of ambiguous location ids: GPS coordinates of 13 location-ids in `FourSquare` corresponds to a single, unique location on GoogleMaps

	Users	Locations	Check-ins	POIs
`FourSquare`	16K	803K	1.928M	582K
`BrightKite`	50K	771K	4.686M	631K
`Gowalla`	99.5K	1.257M	6.271M	1.162M

Table 4: Statistics of datasets: number of users, location, visits and clustered locations/POIs

Since M_1 and H_1 provide equal benefit of 0.9 when combined with H_2, we can randomly choose either M_1 or H_1 as second seed.

6 LBSN Data Analysis

The influence models of Section 3.2 have several parameters to set: τ, ω, and θ. Furthermore, while defining the friendship-based bridging visitors, and influence-friends based bridging visitors, the assumption was made that friends tend to follow friends. Before going to the experiments, in this section, we show how to set the thresholds with reasonable values based on an analysis of the LBSN datasets given in Table 4 and verify and confirm the friendship assumption.

Datasets. We used 3 real-world datasets : `FourSquare` [12], `BrightKite`, and `Gowalla` [5]. Each dataset consists of two parts: the friendship graph and an ordered list of check-ins. A check-in record contains the user-id, check-in time, GPS coordinates of location, and a location-id. The statistics of the datasets are given in Table 4.

Data Prepossessing. The real-life datasets required preprocessing because many locations are associated with multiple location identifiers with slightly different GPS coordinates. Consider, for instance, Figure 4. In this figure, 13 GPS coordinates that appear in the `FourSquare` dataset are shown which correspond to different locations Ids in the dataset, but which clearly belong to one unique location. In order to resolve this issue, we clustered GPS points to get POIs. We used the density-based spatial clustering algorithm [21] with parameters *eps*=10 meters and *minpts*=1 to group the GPS points. New

location Ids were assigned to each cluster for all 3 datasets. The statistics of the new Ids are reported in column POIs of Table 4.

6.1 Setting parameters for the influence models

In order to determine the value of influence window threshold ω, we measured the time difference between consecutive visits of users to distinct locations. The cumulative distribution functions (CDF) for three LBSNs are given in Figure 5. It can be seen that for all LBSNs in our study, 80% of the consecutive activities are performed within 8 hours. After that, there is only a moderate increase in the number of activities with respect to the time interval. Thus, in order to capture only the most common activities, we keep $\omega = 8$. However, it can, of course, be changed if the data distribution is different, or there are different user or application requirements.

Next, we find suitable values for the thresholds of the influence models. In order to do that, we considered all the influence models, i.e., $M_{DA}, M_{DR}, M_{FA}, M_{FR}, M_{IA}$, and M_{IR}, for each pair of locations with at least one bridging visitor. The cumulative distribution functions for each of these numbers are depicted in Figure 6. We can utilize the CDF values for controlling the number of influences in the dataset, and thus also for finding the suitable values for thresholds in our models. The values of the thresholds are an application-dependent choice and can be considered accordingly. For example, if an application requires finding many influential relationships and indirectly many influential and influenced locations, then a lower threshold should be considered and vice versa. In this paper, we consider the top 20% influential relationships among locations for all the models. To do that for each influence model, we consider the CDF value of 0.8 (100%-20%=80%) as its threshold. Therefore, the values of $\tau_{DA}, \tau_{DR}, \tau_{IA}, \tau_{IR}, \tau_{FA}$ and τ_{FR} are $2, 0.6, 4, 0.6, 120$ and 0.6, as shown in Figures 6a, 6b, 6c and 6d, 6e and 6f, respectively.

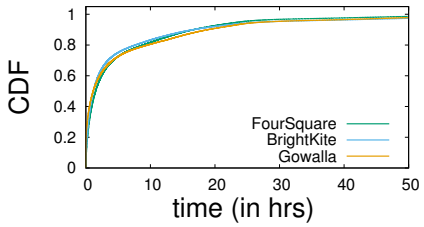
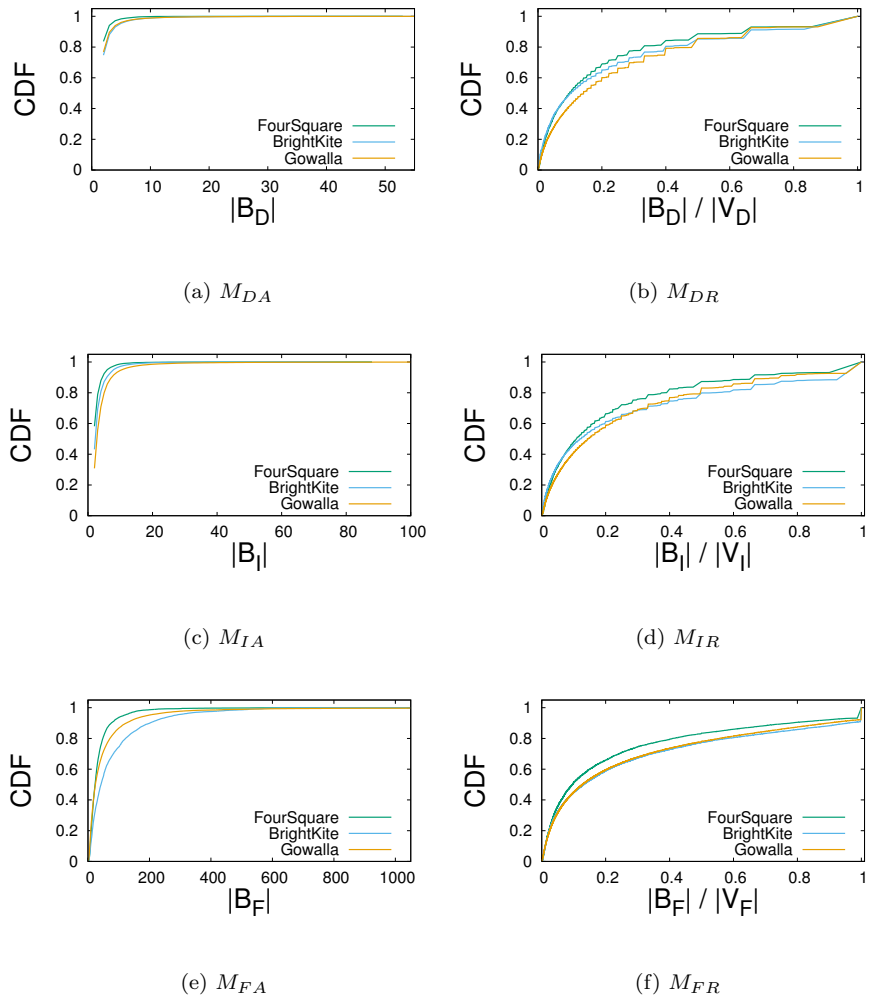


Fig. 5: CDF of time difference between consecutive visits of users to distinct locations



(a) M_{DA} (b) M_{DR}

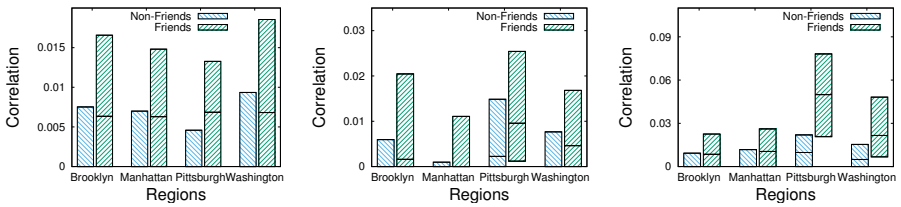
(c) M_{IA} (d) M_{IR}

(e) M_{FA} (f) M_{FR}

Fig. 6: Cumulative distribution function (CDF) of thresholds for corresponding influence models. All locations pairs having at least one bridging visitors are considered and CDF values of the threshold based on the bridging visitors of these locations and visitors of the destination locations are plotted.

6.2 Mobility analysis of friends

In real life, usually activities of friends are more similar than activities of non-friends. In LBSNs, this implies that a visit of a user to a location increases the chances of visits of her friends to the same location. We considered this assumption when constructing our friendship-based bridging visitors and influenced-friends based bridging visitors in Section 3.1.2 and Section 3.1.3, re-



(a) FourSquare (b) BrightKite (c) Gowalla

Fig. 7: Jaccard index based correlations of activities of friends and non-friends users.

spectively. We now show the correctness of this assumption by computing the correlations between activities of users, their friends and non-friends: Let L_u and L_v be the locations visited by users u and v, respectively. The correlation between activities of u and v is measured by the Jaccard Index [3] between L_u and L_v given by $|L_u \cap L_v|/|L_u \cup L_v|$. The average correlation of activities of users and those of their friends is denoted *friendship correlation* (p^f_{corr}), and the average correlation between activities of users and their non-friends is denoted *non-friendship Correlation* (p^{nf}_{corr}). In order to avoid an unreasonable bias due to the fact that friends tend to live in the same city, we restricted our computation of the average non-friendship correlation to users in the same city. We picked four regions of the United States, i.e., Brooklyn, Manhattan, Pittsburgh, and Washington and considered the activities of users in these regions to study the correlations. The statistics of p^f_{corr} and p^{nf}_{corr} of all the users are given in Figure 7. The figure presents boxplots without outliers. It can be seen that the median of p^f_{corr}, even though still small, is up to 5 times larger than of p^{nf}_{corr}. The same pattern is observed for all the datasets. This validates our assumption that the activities of friends are more similar than those of non-friends.

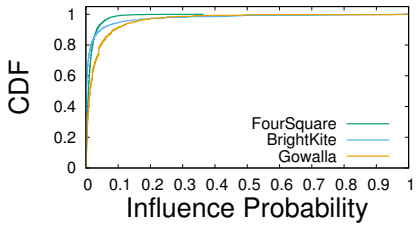


Fig. 8: CDF of Influence Probabilities of all the pairs of influential and influenced users

Although the activities of friends are more correlated than those of non-friends, even between friends the correlation between the sets of locations they visited is low. Thus, in order to tackle this, we considered only the potential influenced friends rather than all friends. To do that we computed the influence probabilities of users among each other using the method given in Section 3.1.3. We plotted the CDF of these influence probabilities and found a suitable value for the influence threshold, as shown in Figure 8. The value of the influence threshold depends on the application. For example, a higher value would be given to θ if a stronger relationship among influential and influenced users is expected. In this paper, we assume that users having influential probabilities in the top 20% will follow the influential users within ω. Thus, by default, we consider $\theta = 0.2$. However, the insights for location influence with respect to different values of θ are also shown in Section 7.2.

7 Evaluation

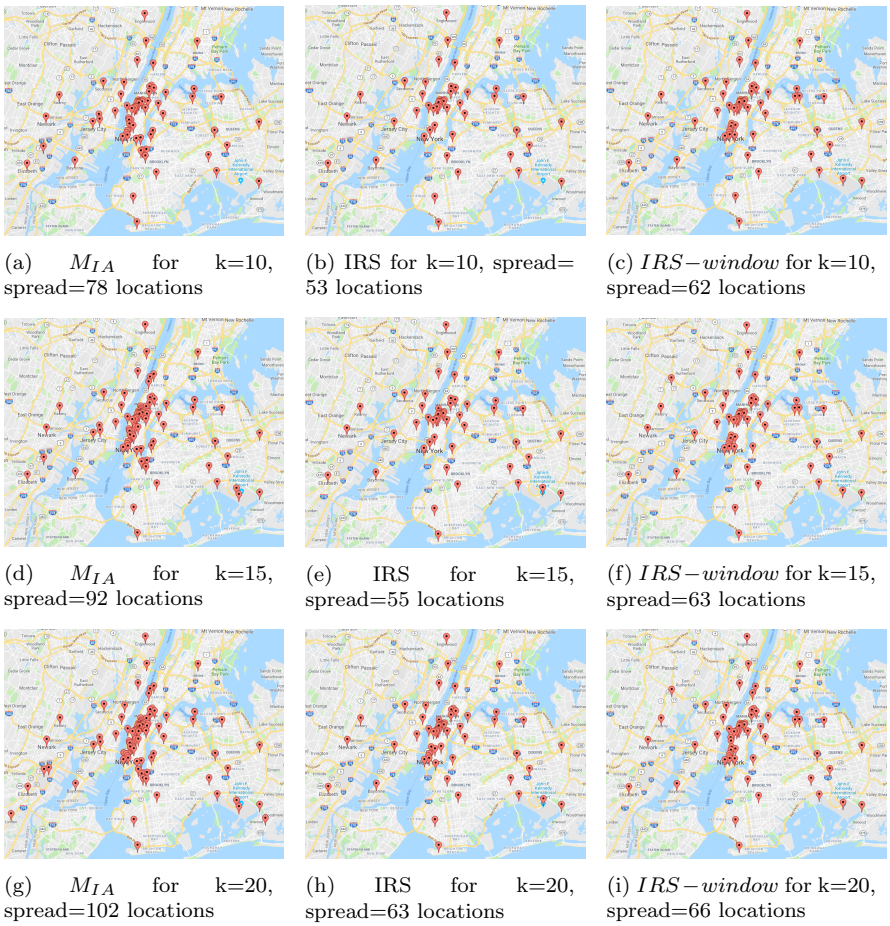
In this section, we evaluate the notions defined in Section 3 and the algorithms introduced in Section 4 and Section 5, respectively.

Experiment settings. We conducted our experiments on a Linux machine with 4 AMD Opteron 6376 processors with 2.3GH and 512 GB RAM. The algorithms [1] are implemented in Scala. The description and preprocessing of the datasets used for the experiments are given in Section 6.

Base-line competitors: To evaluate the effectiveness, we consider the most relevant state-of-the-art influence maximization method, called *Influence Reachability Set (IRS)* [19]. For each node u, IRS fetches all the nodes which are reachable from u based on the temporal path [29] within a given time window. Once the influence reachability sets of all the nodes are obtained, the top-k most influential nodes are found using the standard greedy algorithm such that the combined influence reachability set size of these nodes is the maximum. We consider this algorithm for comparison with our models due to following common features: 1) the model of capturing interactions among users based on their activities, 2) consideration of temporal sliding window for influence estimation, 3) using a standard greedy algorithm for finding top-k influential nodes. The IRS method is used for finding the influence of users among each other based on their activities. However, in our approach, we consider bridging visitors for constructing relationships among locations. In order to evaluate IRS on LBSN data, we propose the two methods below for constructing location interaction graphs. We then run the IRS algorithm for finding top-k influential locations.

– **Location interaction graph:** In this approach, we use the check-in data to generate a graph of interactions between locations. An interaction graph is a graph where the edges between the vertices are timestamped and represents an interaction between the node at that time-stamp. In LBSN data,

[1] Code of the algorithms are given at: https://github.com/rohit13k/LBSNAnalysis



(a) M_{IA} for k=10, spread=78 locations
(b) IRS for k=10, spread= 53 locations
(c) $IRS-window$ for k=10, spread=62 locations
(d) M_{IA} for k=15, spread=92 locations
(e) IRS for k=15, spread=55 locations
(f) $IRS-window$ for k=15, spread=63 locations
(g) M_{IA} for k=20, spread=102 locations
(h) IRS for k=20, spread=63 locations
(i) $IRS-window$ for k=20, spread=66 locations

Fig. 9: Influenced locations w.r.t. to different number of top-k influential locations fetched by M_{IA}, IRS, and $IRS-window$ in NYC for `BrightKite`.

we consider two locations l_1, l_2 are interacting at time stamp t when a user does consecutive check-ins (l_1, t') and (l_2, t). The method of finding top-k influential locations using IRS on location interaction graph is denoted by *IRS*.

– **Time window based location interaction graph:** Usually, influence remains active within a particular time. Hence, we consider this observation for constructing interactions of locations from the LBSN data. We consider an interaction between two locations l_1, l_2 at time stamp t if a user u performs consecutive check-ins (l_1, t') and (l_2, t) such that $t - t' \leq \omega$. The method of finding top-k influential locations using IRS on time window based location graph is denoted by $IRS-window$.

Location correlation graph with PageRank: In this approach, we construct a location correlation graph [26] for a given LBSN. An edge between two locations in a location correlation graph exists if the number of common visitors between them is greater than a threshold. Since, we take the value of $\tau = 2$ for the absolute influence model so, we take the same value, i.e., 2 for the threshold. We then use the Jaccard index to compute the edge weights based on the common visitors and the visitors of the locations. Once the location correlation graph is created, we run the PageRank algorithm to find the top-k locations having the highest PageRank values. This approach is denoted by $PR-LCG$. The idea behind using the location correlation graph is to evaluate the significance of our proposed models for capturing the location influence.

7.1 Qualitative significance of the Location Influence

In order to demonstrate our notion of location influence, we compared the results of our method using the absolute influence friends based bridging visitor model (M_{IA}) with IRS and $IRS - window$. To do that, we considered the activities performed in New York and fetched top-k influential locations for k=10, 15, 20 using M_{IA}, IRS and $IRS-window$. We then plotted the coordinates of the influenced locations of the top-k influential locations using Google Map, as shown in Figure 9. In the figure, it can be observed that M_{IA} leads to a set of locations with a larger spread as compared to the other two approaches, both geographically and in terms of the number of influenced locations. It can further be seen that the difference of influenced locations increases with an increasing value of k. This shows the significance of our proposed method. All the datasets show similar trends. Thus, due to space limits, the results are only shown for the `BrightKite` because of its median data size and check-in density among all three datasets.

7.2 Comparison of Influence Models

Influence spread prediction: Next, we evaluated the influence spread prediction ability of all the models and compared the results with the baseline approaches. To do that, we divided the activities based on time such that each part is composed of the activities of one month. We computed the seeds on one part of the dataset called training set and found their spread on the next part in the sequence called test set. M_A was used to compute the influence on the test set. To compare the results, we computed and compared the number of influenced nodes for the top-k influential locations where k=1, 3, 5, 10, 15, and 20. We iteratively repeated the experiment for all the parts of activities and reported the total spread of all of these iterations for each value of k. The results are shown in Table 5. Here, it can be observed that our proposed models, outperformed both IRS and $IRS - window$ for all the values of k, for all the three datasets. Further, our proposed models also outperformed $PR-LCG$ for

Dataset	K	Number of Influenced Nodes								
		Absolute Influence Model			Relative Influence Model			IRS	IRS-window	PR-LCG
		M_{DA}	M_{FA}	M_{IA}	M_{DR}	M_{FR}	M_{IR}			
FourSquare	1	89	37	**98**	66	66	66	79	71	N/A
	3	190	116	**195**	159	160	159	82	105	
	5	230	117	**255**	216	196	216	121	116	
	10	322	159	**361**	273	260	273	124	167	
	15	380	200	**421**	307	288	307	130	174	
	20	432	255	**495**	330	326	330	132	191	
BrightKite	1	662	657	**671**	648	655	648	512	537	657
	3	943	882	**959**	896	882	896	613	743	924
	5	**1,153**	9,67	1,112	1,041	994	1,041	666	835	1,100
	10	1,458	1,140	**1,465**	1,269	1,259	1,269	749	1,027	1,437
	15	**1,717**	1,257	1,686	1,449	1,444	1,449	821	1,171	1,693
	20	1,921	1,381	**1,951**	1,589	1,570	1,589	867	1,275	1,928
Gowalla	1	676	111	**982**	446	405	446	453	613	N/A
	3	**1804**	153	1787	1129	1212	1129	821	1072	
	5	2834	238	**3087**	1772	1716	1772	997	1134	
	10	4875	860	**5197**	3302	3218	3302	1116	1445	
	15	6236	1395	**6767**	4136	3645	4136	1244	1854	
	20	7460	1877	**8165**	4668	4119	4668	1452	2015	

Table 5: Influence spread of top-k influential locations fetched by the proposed influence models, IRS, $IRS - window$ and $PR - LCG$. The check-ins are divided on monthly basis. The influential keys are fetched on one part and spread is computed on the next part in the sequence. The process is iteratively repeated for all the months and the total influence spread for each value of k, for each dataset is depicted.

BrightKite. We only computed $PR - LCG$ for BrightKite due to intensive computation time, i.e., more than 48 hours, required for constructing the location correlation graph. For all the methods, the difference in spread increases with an increasing value of k. Overall, the top-k influential locations fetched by the absolute influence models influenced more locations as compared to the relative influence models. More specifically, the spread of M_{IA} is up to 45% more than M_{DA}, 700% more than M_{FA}, and 400% more than $IRS - window$. The reason is that unlike the M_{DA} and M_{FA}, M_{IA} incorporate the friends of bridging visitors that have potential to become bridging visitors in future. The relative models filter out the excessive influence of popular locations thus the influence spread of the models is almost same.

Computational resources. We compared the computation time and memory requirements for the influence models. The results are shown in the Table 6. Here, it can be observed that the computation time for M_{IA} is more than M_{DA}. The reason is that for the influenced-friends based bridging visitors, we further need to compute the influence probabilities of users among each other which requires more computation time. Since we incorporate the

Dataset	Time			Memory		
	M_{DA}	M_{FA}	M_{IA}	M_{DA}	M_{FA}	M_{IA}
FourSquare	596	2,220	1,140	29,562	77,334	29,654
BrightKite	808	4,363	1,259	30,199	228,391	30,602
Gowalla	2,957	Out of Memory	6,973	186,363	Out of Memory	189,195

Table 6: Computation time and memory for all influence models.

		FourSquare			Gowalla		
		mean $\pm\sigma$			mean $\pm\sigma$		
		b=64	b=128	b=256	b=64	b=128	b=256
Rel. error	M_{DA}	0.03 ± 0.15	0.01 ± 0.01	0.01 ± 0.06	0.03 ± 0.17	0.01 ± 0.12	0.01 ± 0.01
	M_{FA}	0.26 ± 0.45	0.19 ± 0.34	0.15 ± 0.35	Out of Memory		
	M_{DR}	0.05 ± 0.21	0.05 ± 0.21	0.05 ± 0.2	0.17 ± 0.41	0.17 ± 0.41	0.17 ± 0.4
	M_{FR}	0.05 ± 0.21	0.05 ± 0.21	0.05 ± 0.21	Out of Memory		

Table 7: Accuracy (relative error) for approximate algorithms w.r.t bucket size.

influence-friends of the bridging visitors as well for constructing influence oracle thus more memory is consumed. The computation time and memory consumption of M_{FA} is more than both other models. The reason is that for friends-based bridging visitors, we incorporate all the friends of bridging visitors which largely increases the bridging visitor set. Thus, it requires more time to process and more memory to maintain the set for constructing influence oracle. The base line competitors, $IRS, IRS-time$ and $PR-LCG$ require more computational resources in comparison with all the influence models. The reason is that an intensive computation is required for constructing the location interaction graph, the time window based location interaction graph, and the location correlation graph for $IRS, IRS-time$, and $PR-LCG$, respectively.

Conclusion: Our proposed models outperformed baseline competitors in terms of influence spread as well as required computational resources. Overall, the absolute models performed better than the relative models. In case of the relative models, M_{DR} should be chosen as it needs minimum computational resources but yields the same influence as other models. For the absolute models, although M_{IA} requires more computational resources, but yields the maximum influence. On the other hand, M_{DA} requires fewer resources but the influence spread is also lower. M_{FA} is the worst in terms of computational resources and influence spread. Thus, the choice of the model among M_{IA} and M_{DA} can be made by considering the trade-off between computational resources and influence spread, i.e., if a higher influence spread is more desirable than computational resources then M_{IA} should be chosen, and vice versa.

7.3 Approximations for M_{DA} and M_{FA}

Next, we analyzed the approximate algorithms for constructing influence oracle for M_{DA} and M_{FA}. We analyzed the impact of approximation on accuracy, computation time and memory. The results are similar for all the datasets and hence we only present results for the smallest and the biggest datasets, i.e., `FourSquare` and `Gowalla`, respectively.

Approximation accuracy. For every location with a non-empty influence set, we used the HLL-based approximate version of the Oracle to predict the size of the influence set. Then the relative error as compared to the real size was computed for every location. In Table 7 the mean and standard deviation of this relative approximation error over all locations with a non-empty influence

are given. The experiments are performed for M_{DA}, M_{FA}, M_{DR}, and M_{FR}. We ran the experiments for different numbers of buckets (b) for the *HLL* sketch, being, 64, 128 and 256. It can be seen in the table that the errors are unbiased (0 on average) and the standard deviation decreases as the number of buckets increases. The error is a bit higher for M_R as compared to M_A because in the relative model the influence is computed by taking the ratio of two approximated set cardinalities. Values for b beyond 256 yielded only modest further improvements and hence we used $b = 256$ in all further experiments. The results for M_{FA} and M_{FR} for the `Gowalla` could not be computed due to the huge memory requirement.

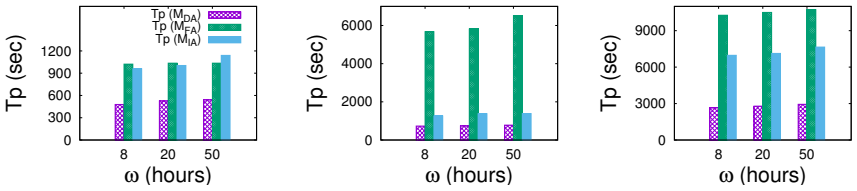
Time and space consumption. Next, we analyzed the computation time and memory requirements for the approximate approach by computing the influence oracles for M_D and M_F. The results are shown in Table 8. It can be observed that time and memory increase with increasing number of buckets b. Furthermore, it can also be observed that the approximate approach outperformed the exact approaches in computation time and memory given in Table 6. The improvement for M_F in the computation time is two folds while using only 18% of memory. Due to the sparsity of data, however, the gain for M_D is less, i.e., 63% of the time and 48% of the memory is required by the approximate approach as compared to exact one. This is because the sizes of the sets of bridging visitors are very modest. The results for `Gowalla` for M_F could not be computed by the exact algorithm due to insufficient memory. However, for all the bucket sizes the approximate algorithm computes it by taking less than half of the available memory.

7.4 Effects of parameters: ω and τ

Computation time. We studied the runtime of the algorithms on all the datasets for different values of $\omega := 8, 20$ and 50. Considering the significant improvement of the approximate algorithms in computational resources without compromising on accuracy, for all further experiments, we use approximate variants for M_{DA} and M_{FA}. However, the exact algorithm for M_{IA} is used as we do not have its approximate variant. The average runtime for processing all the activities (T_p) under the models varies only depending on the influence models, i.e, whether or not we consider friends; it does not depend on τ. Also, the oracle query time (T_q) is independent of τ and the influence model. The

		FourSquare			Gowalla		
		b=64	b=128	b=256	b=64	b=128	b=256
Time	M_{DA}	378	428	544	2,031	2,605	2,499
	M_{FA}	928	955	993	9,479	9,685	9,912
Memory	M_{DA}	14,275	18,636	27,372	89,418	116,847	171,908
	M_{FA}	14,569	18,943	27,726	90,793	118,173	172,985

Table 8: Time (sec) and memory (MB) required by the approximate algorithms w.r.t, bucket size.

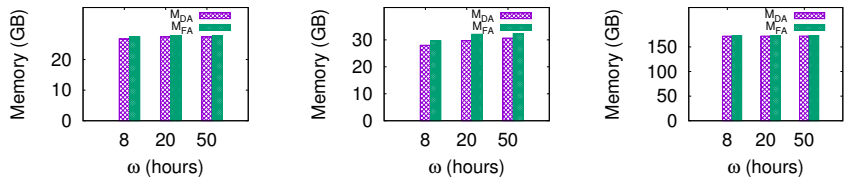


(a) `FourSquare` (b) `BrightKite` (c) `Gowalla`

Fig. 10: Total Time (Tp) in seconds w.r.t. ω to process all activities for $\tau = 2$ under M_{DA}, M_{FA} and M_{IA} models.

run times for processing all the activities are shown in Figure 10 for the three datasets `FourSquare`, `BrightKite` and `Gowalla`. The running time increases with increasing influence window size ω as more locations from the visit history remain active. Running time is higher in the M_F which is not surprising either as the number of users to include in the bridging visitors sets increases due to the addition of friends. For the M_I, we needed to find influenced users of bridging visitors thus computation time is higher as compared to M_D. The time taken to process `Gowalla` dataset is the highest as it has the largest number of locations.

Memory consumption. We also studied the memory required by the approximation algorithm on all the datasets for different values of $\omega := 8, 20$ and 50. Unlike for the processing time, the average memory required to process all the activities under M_D and M_F does not vary based on whether we consider friends or not. This is because the HLL sketch storing the bridging visitor set size remains constant in size even if a larger number of users is added to it. The memory requirement increases slightly with ω as more locations are getting influenced due to a larger influence window. The results are shown in Figure 11. The total memory requirements increased linearly with time as new locations came in over time for which a complete influence summary was needed to be maintained. We further pruned the outdated locations in the visit histories thus, over time the size of user history remained constant as shown in Figure 12.



(a) `FourSquare` (b) `BrightKite` (c) `Gowalla`

Fig. 11: Total Memory in GB w.r.t. ω to process all activities for $\tau = 2$ under M_{DA} and M_{FA}.

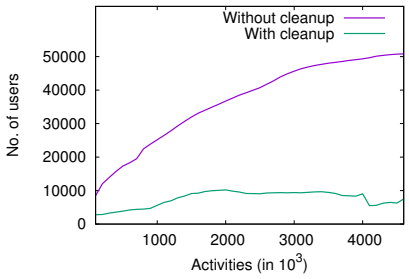
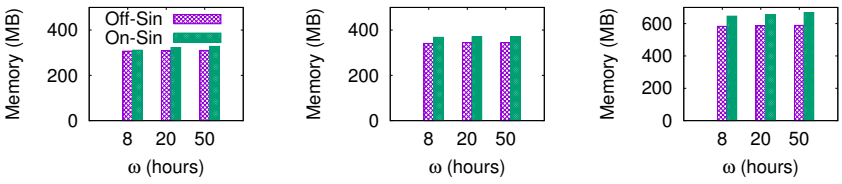


Fig. 12: User visit history set size growth w.r.t to number of activities processed with and without cleanup process.

7.5 Single-Influencer based Location Influence

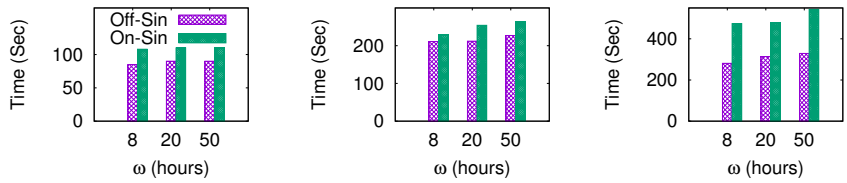
Next, we present the resource requirements of the algorithms for constructing single-influencer based influence oracles. The memory consumption by the *On-Sin* and the *Off-Sin* algorithms is shown in Figure 13. Since the memory requirement for the algorithms is $log(\omega)$, thus a very modest change in the memory with respect to different values of ω was observed. The *Off-Sin* requires less memory as compared to the *On-Sin*. The reason is that for the *Off-Sin*, we consider the activities in reverse order of time, so we approximate the user history using the probabilistic data structure. However, for *On-Sin*, we maintain the exact user history which requires more memory. The difference of time and memory requirements among *On-Sin* and *Off-Sin* is modest because of the data sparsity and small window size. However, these algorithms for the special case outperformed the approximate algorithm for all influence models (shown in Figure 11) up to 22x in memory consumption. The reason is for the special case we do not maintain the bridging visitor set because the influence is spread by a single carrier, and thus the memory is saved.

The computation time by the *On-Sin* and *Off-Sin* are shown in Figure 14. Here, it can be observed that for the *Off-Sin* algorithm, like for memory consumption, there was a modest improvement in computation time as compared to the *On-Sin*. Similarly, both of these algorithms required up to 20x less



(a) `FourSquare` (b) `BrightKite` (c) `Gowalla`

Fig. 13: Comparison of memory consumption for *Off-Sin* and *On-Sin*



(a) `FourSquare` (b) `BrightKite` (c) `Gowalla`

Fig. 14: Comparison of computation time for *Off-Sin* and *On-Sin*

computation time as compared to the approximate algorithm for all influence models (shown in Figure 10).

7.6 Influence Maximization

Influence of α. Our next goal is to study how the influence maximization algorithm performs for different values of α. In order to avoid data sparsity issues, we filtered out those locations which have only one visitor from all the datasets. We tested the spread of top 200 locations obtained by considering values of α from 0.01 to 0.99. We observed that the number of bridging visitors per location is highly skewed as can be learned from Figure 6a. Due to this, the potential influenced locations having few bridging visitors are less likely to affect the influenced set of the locations. The effect of varying alpha on the influence spread is shown in Figure 15. As expected for these sparse datasets, our algorithms performed best with a lower value of α. We use $\alpha = 0.03$ for our experiments.

Computation time. We study the computation time for finding top-k influential locations under all influence models. The runtime is close in the both M_A and M_R. The time increases with k. Nevertheless, the increase is modest; for instance, finding the top-50 locations takes less than 2 minutes for `FourSquare`. We report the results in Table 9.

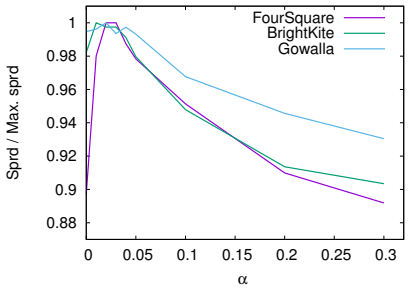


Fig. 15: Plot of ratio of total influence spread w.r.t. alpha (200 seeds).

τ	Time (sec)								
	FourSquare			BrightKite			Gowalla		
	$k = 10$	$k = 20$	$k = 50$	$k = 10$	$k = 20$	$k = 50$	$k = 10$	$k = 20$	$k = 50$
$\tau_{DA} = 2$	37	40	120	24	50	218	132	213	723
$\tau_{DR} = 0.6$	108	101	180	84	132	545	521	525	778
$\tau_{IA} = 5$	10	18	60	15	32	208	70	173	706
$\tau_{IR} = 0.6$	50	45	99	37	73	177	292	346	563
$\tau_{FA} = 120$	19	61	525	35	553	7776	181	1168	19302
$\tau_{FR} = 0.6$	68	96	591	201	151	591	450	557	1386

Table 9: Time taken to find top k locations.

8 Conclusion and Future Work

In this paper, we introduced a location influence maximization approach that can be used to optimize outdoor marketing strategies such as finding optimal locations for advertising products to maximize the geographical spread. In order to do that, we captured the interactions of locations on the basis of their visitors to compute the influence of locations among each other. We provided two models namely the absolute influence model and the relative influence model. We further provide three variants of these models that incorporate the social graph and consider the friends of users that have potential to repeat their activities in future, and improve the location influence up to 45%. We proposed a data structure: influence oracle to efficiently compute the influence of locations on the basis of these models for finding top-k influential locations. In order to maintain this data structure, we first provided a set-based exact algorithm. Then, we optimized the time and memory requirements of the algorithms by utilizing a probabilistic data structure. We further introduced a method in which single carriers can be used to spread the influence. For this case, we provide two algorithms, an off-line and an on-line. With the help of these algorithms, we further improved the computation time and memory requirement by 20x and 22x, respectively. Finally, we provided a greedy algorithm to compute the top-k influential locations. In order to evaluate the methods, we utilized three real datasets. We first analyzed the LBSN datasets: `FourSquare`, `BrightKite` and `Gowalla` to verify some claims and to provide optimal values for thresholds of the influence models. Then, we evaluated our approaches for the computation of the Oracle and finding top-k locations in terms of accuracy, computation time, memory requirement and scalability. We further show the effectiveness of our proposed models by comparing the influence spread of top-k locations fetched by our approach with that of two variants of a state-of-the-art approach; IRS and IRS-window.

In the future, we plan to enrich location influence models by incorporating the activities users perform with their friends in groups. Moreover, we aim to provide distributed techniques for computing the Oracle data structures and influences for the models.

Acknowledgments

This research has been funded in part by the European Commission through the Erasmus Mundus Joint Doctorate "Information Technologies for Business Intelligence - Doctoral College" (IT4BI-DC). Rohit Kumar is supported by Fonds de la Recherche Scientifique-FNRS under Grant n^o T.0183.14 PDR.

References

1. A. AlDwyish, E. Tanin, and S. Karunasekera. Location-based social networking for obtaining personalised driving advice. In *SIGSPATIAL*, 2015.
2. P. Bouros, D. Sacharidis, and N. Bikakis. Regionally influential users in location-aware social networks. In *SIGSPATIAL*, 2014.
3. R. R. Braam, H. F. Moed, and A. F. Van Raan. Mapping of science: Critical elaboration and new approaches, a case study in agricultural biochemistry. In *Informetrics*, 1988.
4. W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD*, 2009.
5. E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *KDD*, 2011.
6. E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *CIKM*, 2014.
7. T.-N. Doan, F. C. T. Chua, and E.-P. Lim. Mining business competitiveness from user visitation data. In *SBP*, 2015.
8. P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, 2001.
9. N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha. Scalable influence estimation in continuous-time diffusion networks. In *NIPS*, 2013.
10. L. Ferrari, A. Rosi, M. Mamei, and F. Zambonelli. Extracting urban patterns from location-based social networks. In *SIGSPATIAL*, 2011.
11. P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *DMTCS*, 2008.
12. H. Gao, J. Tang, and H. Liu. Exploring social-historical ties on location-based social networks. In *AAAI*, 2012.
13. M. Gomez-Rodriguez and B. Schölkopf. Influence maximization in continuous time diffusion networks. In *ICML*, 2012.
14. A. Goyal, F. Bonchi, and L. V. Lakshmanan. Discovering leaders from community actions. In *CIKM*, 2008.
15. A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *WSDM*, 2010.
16. A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. A data-based approach to social influence maximization. In *PVLDB*, 2011.
17. N. T. Hai. A novel approach for location promotion on location-based social networks. In *RIVF*, 2015.
18. D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.
19. R. Kumar and T. Calders. Information propagation in interaction networks. In *EDBT*, 2017.
20. G. Li, S. Chen, J. Feng, K.-l. Tan, and W.-s. Li. Efficient location-aware influence maximization. In *SIGMOD*, 2014.
21. Q. Liu, M. Deng, Y. Shi, and J. Wang. A density-based spatial clustering algorithm considering both spatial proximity and attribute similarity. In *Computers and Geosciences*, 2012.
22. L. LováSz. Review of the book by alexander schrijver: Combinatorial optimization: Polyhedra and efficiency. In *Oper. Res. Lett.*, 2005.

23. F. J. Mata and A. Quesada. Web 2.0, social networks and e-commerce as marketing tools. In *J. Theor. Appl. Electron. Commer. Res.*, 2014.
24. M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, 2002.
25. M. A. Saleem, R. Kumar, T. Calders, X. Xie, and T. B. Pedersen. Location influence in location-based social networks. In *WSDM*, 2017.
26. M. A. Saleem, X. Xie, and T. B. Pedersen. Scalable processing of location-based social networking queries. In *MDM*, 2016.
27. X. Wang, Y. Zhang, W. Zhang, and X. Lin. Distance-aware influence maximization in geo-social network. In *ICDE*, 2016.
28. Y.-T. Wen, P.-R. Lei, W.-C. Peng, and X.-F. Zhou. Exploring social influence on location-based social networks. In *ICDM*, 2014.
29. H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu. Path problems in temporal graphs. *Proceedings of the VLDB Endowment*, 7(9):721–732, 2014.
30. H.-H. Wu and M.-Y. Yeh. Influential nodes in a one-wave diffusion model for location-based social networks. In *PAKDD*, 2013.
31. C. Zhang, L. Shou, K. Chen, G. Chen, and Y. Bei. Evaluating geo-social influence in location-based social networks. In *CIKM*, 2012.
32. T. Zhou, J. Cao, B. Liu, S. Xu, Z. Zhu, and J. Luo. Location-based influence maximization in social networks. In *CIKM*, 2015.
33. W.-Y. Zhu, W.-C. Peng, L.-J. Chen, K. Zheng, and X. Zhou. Modeling user mobility for location promotion in location-based social networks. In *KDD*, 2015.